

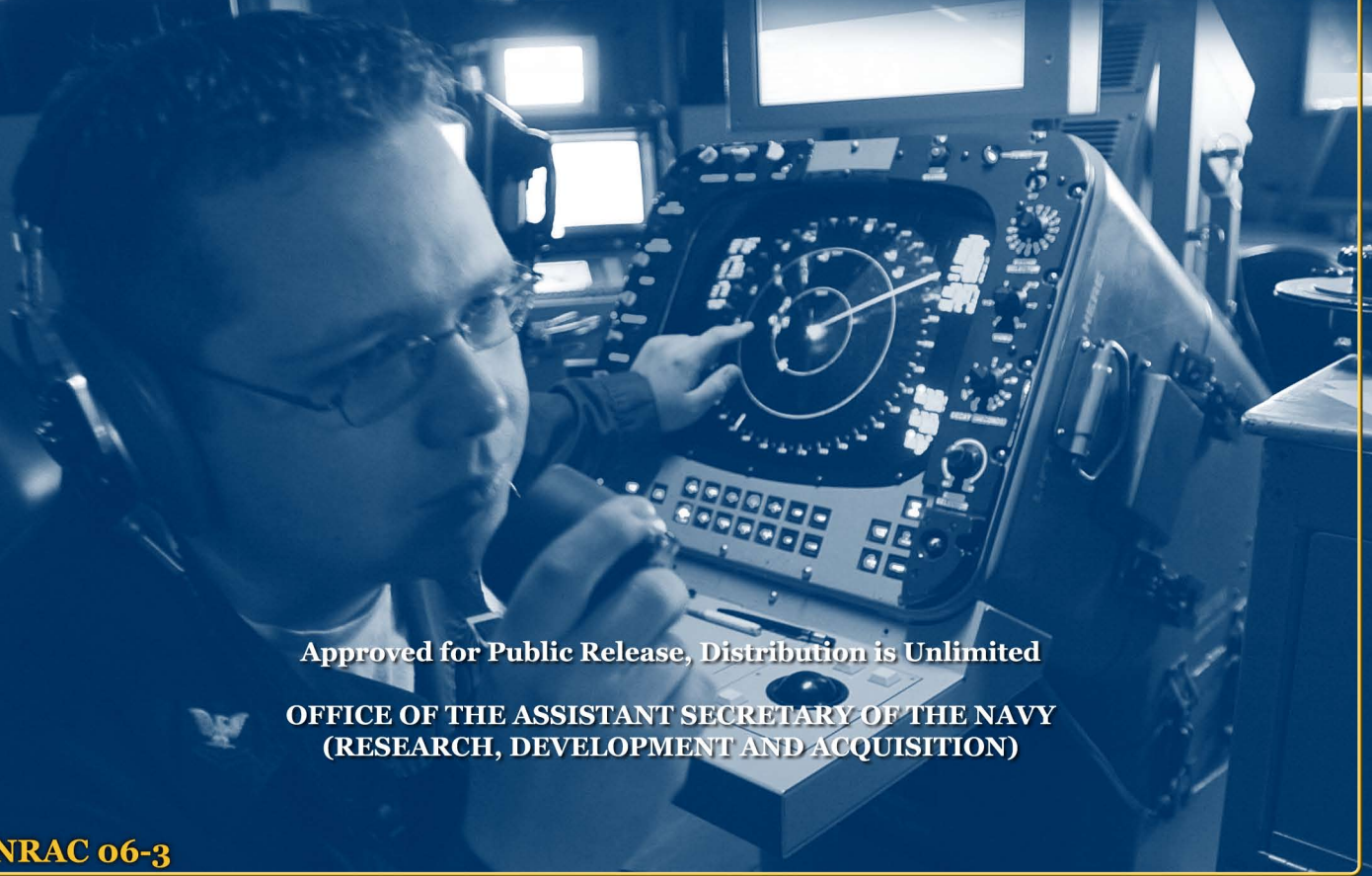


Naval Research Advisory Committee Report



Software Intensive Systems

July 2006



Approved for Public Release, Distribution is Unlimited

OFFICE OF THE ASSISTANT SECRETARY OF THE NAVY
(RESEARCH, DEVELOPMENT AND ACQUISITION)

This report is a product of the United States Naval Research Advisory Committee (NRAC) Panel on SOFTWARE INTENSIVE SYSTEMS. Statements, opinions, recommendations, and/or conclusions contained in this report are those of the NRAC Panel and do not necessarily represent the official position of the United States Navy and United States Marine Corps, or the Department of Defense.

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 0704-0188	
The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.					
1. REPORT DATE (DD-MM-YYYY) 03/01/2006		2. REPORT TYPE Group Study		3. DATES COVERED (From - To) March 2006- July 2006	
4. TITLE AND SUBTITLE Software Intensive Systems				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S) E. Horvitz, D.J. Katz, R.L. Rumpf, H. Shrobe, T.B. Smith, G.E. Webber, W.E. Williamson, P.H. Winston, James L. Wolbarsht				5d. PROJECT NUMBER	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Research Advisory Committee 875 Randolph St. Arlington, VA 22203-1993				8. PERFORMING ORGANIZATION REPORT NUMBER NRAC 06-3	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Assistant Secretary of the Navy (Research, Development and Acquisition) 1000 Navy Pentagon Washington, DC 20350-1000				10. SPONSOR/MONITOR'S ACRONYM(S) ASN(RD&A)	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT DISTRIBUTION STATEMENT: Approved for public release; distribution is unlimited.					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT Consistent production of quality, affordable software. Other countries have surpassed the US in computer manufacturing and much software production is now outsourced abroad. Recommend that DoN create a software acquisition specialty, mandate basic schooling for software acquisition specialists, close certain acquisition loopholes that permit poor development practices, and promote the careful use of existing technology and the development of gap-filling technology. Additionally, recommend that DoN invest in software engineering, particularly as it complements commercial industry developments and promotes the application of systems engineering methodology. Central recommendation is a three-step mobilize-transform-consolidate process, starting with project-directed RESET teams (Rapid Evolution of Software Engineering Technology) inserted on-site at contractor locations, continuing with a development of Naval Software System Center, and evolving into a larger Naval software organization.					
15. SUBJECT TERMS Software, RESET, globalization, FORCEnet, acquisition, training, computers, Ultra large systems, Unified Modeling Language, Single integrated air picture, Single lines of code, MSLOC, MDA, MDD, UARC					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. Sujata Millick
UNCLAS	UNCLAS	UNCLAS	UU	86	19b. TELEPHONE NUMBER (Include area code) (703) 696-4875

This page intentionally left blank



Naval Research Advisory
Committee Report



Software Intensive Systems

July 2006

Approved for Public Release, Distribution is Unlimited

**OFFICE OF THE ASSISTANT SECRETARY OF THE NAVY
(RESEARCH, DEVELOPMENT AND ACQUISITION)**

This page intentionally left blank

Table of Contents

Executive Summary	3
Terms of Reference	5
Study Panel and Sponsor	7
Briefings and Visits.....	9
Joint Vision	13
More Capability and Lower Cost.....	15
The Playing Field.....	17
Human Resources	19
Globalizing of Software and Hardware	23
Spending	25
Impact of Rework Costs (FY 2003).....	27
Size of Typical Naval Combat Systems.....	29
History of Study.....	31
Our Central Recommendation: Structural Innovation	33
Representative Findings.....	35
Leadership Recommendations.....	41
Acquisition and Practice Recommendations	43
Recommendation Focus: The User Requirement Loop.....	49
Naval S&T Program Recommendations.....	51
Assessment.....	55
Step One: Rapid Evolution Software Engineering Teams.....	57
Step One: Implementation	59
Step Two: Naval Software System Center.....	61
Step Two: Implementation.....	63
Step Three: Consolidation.....	65
Risks and Challenges: Steps One-Three	67
Summary	69
Terms of Reference	A-1
ASN Memo	B-1
Acronym List	C-1

Executive Summary

Purpose of study

The consistent production of quality, affordable software has become enormously important, because information dominance has become a cornerstone of national defense. Unfortunately, abundant examples demonstrate that DoN and DoD software programs are often late, over budget, and under performing.

All this is especially worrisome in light of the globalizing of competence in information technology. Other countries have surpassed the United States in computer manufacturing and much software production is now outsourced abroad.

On the other hand, emerging technologies and processes, such as model-driven design and software product lines, offer promise, inviting development and selected deployment. The purpose of this study was to assess the state of the art in software development, offer deployment suggestions, and identify S&T needs and opportunities.

Findings and general recommendations

Overall, we saw great benefits to be gained from emerging practices; including lower cost, greater security, more reliability, increased interoperability, easier maintenance, better compliance with requirements, more agile evolution, and more openness.

Such benefits led us to several recommendations focused on DoN acquisition management, systems engineering, training, education, and business practices. In the small, we recommend that the DoN create a software acquisition specialty, mandate basic schooling for software acquisition specialists, close certain acquisition loopholes that permit poor development practices, and promote the careful use of existing technology and the development of gap-filling technology.

We also recommend DoN investment in software engineering, particularly as it complements commercial industry developments and promotes the application of systems engineering methodology. Such investment is needed, even though there is a great deal of commercial investment, because much commercial practice requires adaptation before it is useful to the DoN. For example, the practice of the nightly build, commonplace in industry, is relatively rare in the DoN because the real-world exercise of a current prototype, all the way out to firing, say, a missile, is impractical.

As for emerging software acquisition tools for specifying, bidding, and engineering software-intensive systems, we found promise but not full maturity. Specifying and bidding tools are emerging but none are mature enough to seriously evaluate. On the other hand, some software engineering tools are ready for use in selected applications as long as they are matched to problems. Caveat emptor, however, as there is much zealotry out there, as well as misleading claims, such as for automated code generation where claims are made that “no coding is done.” In reality, tools associated with such claims depend on users writing code in so-called action languages to define semantics and specify procedures. Worse yet, users often find themselves forced to escape to a traditional programming language such as C or C++ to fully define necessary semantics.

The key recommendation: a three-step plan

In the large, our central recommendation is a three-step mobilize-transform-consolidate process, starting with project-directed RESET teams (Rapid Evolution of Software Engineering Technology) inserted on-site at contractor locations, continuing with a development of Naval Software System Center, and evolving into a larger Naval software organization. These steps will help suffuse the DON with today's best tools and practices and those tools and practices that emerge going forward.

Missions of the RESET teams are:

- Complete user-requirements loop
- Promote use of system engineering tools, policies and practices
- Champion best-practice software methodology emphasizing commonality, evolution, adaptation, reuse, reliability, interoperability, security, and rapid response to changing defense needs
- Identify open systems needs and ensure compliance
- Recommend contract incentives
- Monitor progress and sustain support

Missions of the Naval Software System Center are:

- Institutionalize and staff RESET teams
- Build models and assist in building models
- Ensure maximum DoN commonality
- Manage and staff Independent Expert Reviews
- Recommend acquisition policy
- Manage innovation through programs, such as SBIRs.

Embarking on this three-step plan involves risks and challenges; but, because the DoN spends on the order of 1.7B/year on software rework, there is ample opportunity for a huge return on investment. If by step two of the plan, the Navy saves just 10% of that rework cost, the plan will have paid for itself 10 times over.



The Terms of Reference

Background

Context

Structure

Findings

Rcmds

Three steps

Summary

- Review relevant DOD and government programs
- Review industry tools, practices, and standards
- Identify potential benefits of best practices
- Recommend changes in Naval acquisition management, systems engineering, training, education, and business practices
- Suggest S&T investment
- As appropriate, evaluate emerging tools for specifying, bidding, and engineering software-intensive systems and suggest strategies for use across multiple organizations



The Terms of Reference

The Terms of Reference asked us to review relevant DoD and government programs, which led us to absorbing briefs from the programs listed on the Briefings slide. Those briefings enabled us to review a variety of much-talked about tools, practices and standards, including those associated with labels such as Product Lines, Model Driven Development, Service Oriented Architectures and the like.

Overall, we saw great benefits from emerging practices, including lower cost, greater security, more reliability, increased interoperability, easier maintenance, better match to requirements, more agile evolution, and more openness.

Such benefits led us to several suggestions focused on DoN acquisition management, systems engineering, training, education, and business practices. In the small, we recommend that the DoN create a software acquisition specialty, mandate basic schooling for software acquisition specialists, close certain acquisition loopholes that permit poor development practices, and promote the careful use of existing technology and the development of gap-filling technology. In the large, our central recommendation is a three-step process, starting with project-directed rapid evolution teams, continuing with a transformation center, and evolving into something larger. If implemented, these actions will help suffuse the DoN with the best of today's practices and those that emerge going forward.

We also recommend DoN investment in software engineering, particularly as it complements industrial developments and promotes the application of systems engineering methodology. Such investment is needed, even though there is a great deal of commercial investment, because much commercial practice requires adaptation before it is useful to the DoN. For example, the practice of the nightly build, commonplace in industry, is relatively

rare in the DoN because for many DoN systems, the real-world exercise of these builds is impractical (such as the firing of a missile)



Study panel and sponsor

Background

Context

Structure

Findings

Remds

Three steps

Summary

- Chair - Dr. Patrick L. Winston
Professor of Computer Science,
MIT
- Co-Chair - Ms. Teresa B. Smith
Director Strategy, SD&T, Northrop
Grumman Electronic Systems
Sector
- Dr. Eric Horvitz
Principal Researcher and Research
Area Manager, Microsoft
- VADM Douglas J. Katz
USN (Ret.), Consultant
- Mr. Richard L. Rumpf
President Rumpf Associates
International
- Dr. Howard Shrobe
Principal Research Scientist, MIT
- Dr. George E. Webber
Consultant
- Dr. Walton E. Williamson, Jr.
Professor and Chair Department of
Engineering Texas Christian
University
- Mr. James L. Wolbarsht
President & CEO, DEFCON®, Inc.

Study Sponsors:

RADM Michael Frick - PEO-IWS
Mr. Carl Siel - CHENG

Executive Secretaries:

Dr. William Bail, MITRE
Ms. Cathy Ricketts, PEO-IWS
Mr. Fred Heinemann, EDO



Study panel and sponsor

The Panel included experts in the subject matter, the defense industry, commercial practice, and the DoN. Some of these were NRAC members and NRAC associates; others were brought onto the panel to complete the set of desired competencies. The three executive secretaries of the Panel provided insight based on their long history of involvement in DoN work.

This page intentionally left blank



Briefings and visits

Background	- Briefings, programs and defense industry - Naval Focus: PEO-IWS; DASN-IWS; LMRS; Aegis; DD(X); FORCEnet; ARCI - Army Focus: FCS, SW Improvement Program (Bolton) - Joint Focus: SIAP, JSF; JTRS; GIG - OSD/Agency Focus: Missile Defense Agency, NSA, Quadrennial Defense Review, NII/GIG
Context	
Structure	
Findings	
Rcmds	- Other briefings - Government: GSA - FFRDC: SEI - Industry: Raytheon, Microsoft, Lockheed Martin
Three steps	
Summary	- Site visits: - SIAP Program Office - GIG Testbed (JHU/APL) - Microsoft Corporation



Briefings and visits

We heard many briefs that collectively provided insight into problems and opportunities. These briefs were provided by representatives from the DoN, other services, other government agencies, the defense industry, commercial industry, the Software Engineering Institute and the MITRE Corporation. (The later two both being Federally Funded Research and Development Centers (FFRDC)). The list of presentations follows:

Commercial Practices Presentations

- Systematic Software Development at Microsoft--Jim Larus, Research Area Manager, Microsoft Research
- Formal Specification and Program Analysis Tools--Manuvir Das, Researcher, Center for Software Excellence
- Static Analysis of Device Drivers via Software Model Checking--Tom Ball, Principal Researcher, Microsoft Research
- The Spec# Programming System--Rustan Leino, Senior Researcher, Microsoft Research
- Singularity--Galen Hunt, Principal Researcher, Microsoft Research
- Model Based Testing--Wolfram Schulte, Research Area Manager, Microsoft Research
- Windows Vista Engineering: Delivering a High Quality OS--Amitabh Srivastava, Corporate Vice President, Windows Core OS
- Zap – Automated Theorem Proving for Program Analysis--Madan Musuvathi, Researcher, Microsoft Research

- Software Factories--Jack Greenfield, Software Architect, Visual Studio, US-Enterprise Tools Management
- SDL and Common Criteria Discussion--Eric Bidstrup, Group Manager, US Security Engineer and Communications
- SOA and Web 2.0 Discussion--Harry Pierson, Architect, Developer and Platform Evangelism
- Microsoft Dynamic Systems Initiative--John Wilson, Architect Windows Management
- Windows Lifecycle--Chris Lindstrom, Group Program Manager, Windows Fundamental Practices
- Productivity Visions: The Microsoft Center for Information Work--Apollo Fuhrman, Tour Host, Microsoft Center for Information Work
- MDA at Raytheon for Real-Time Systems--Terri Potts, Raytheon

Government Program Presentations

- Aegis--Reuben Pitts & CDR John Ailes, Program Executive Office, Integrated Warfare Systems
- Long Term Mine Reconnaissance (LMRS)--CAPT Paul Imes
- Joint Tactical Radio System (JTRS)--Richard North, JPEO JTRS & Leonard Schiavone, MITRE
- Single Integrated Air Picture (SIAP)--CAPT Jeff Wilson
- Single Integrated Air Picture (SIAP) and Model Driven Architecture--Dr. Michael Bienvenu, MITRE, Technical Lead of SIAP Architecture Maintenance
- Single Integrated Air Picture (SIAP) Site Visit, Crystal City--CAPT Jeff Wilson, JSSEO
- GIG Infrastructure--Ken Schmidt/John Piorkowski, APL/JHU
- Navy Open Architecture--CAPT James Shannon, Program Executive Office Integrated Warfare Systems
- Missile Defense Agency – Advanced Battle Manager / Global Integrated Fire Control (ABM/GIFC)--Dr. Butch Caffell
- Acoustic Rapid COTS Insertion / Advanced Processor Build (ARCI/APB)--Dr. Bob Zarnich
- Future Combat System (FCS)--LT COL Dave Basset, PM SW Integration
- Future Combat System (FCS)--Mr. Dave Emery & Mr. Bell
- DD1000--CAPT. Syring
- Joint Strike Fighter (JSF)--Glenn Willis, Capt. Hambli, JSF Program Office,

- FORCEnet Overview--Craig Madsen/Tech Dir SPAWAR 05
- Service Orientation: An Enabler of Net Centric Operations--Brad Mercer, MITRE, Lead Enterprise Systems Architect for Future FORCEnet
- Army Ultra Light Systems (ULS)--Dr. Jim Linnehan
- GIG Test Bed at APL/JHU--Robert Holland

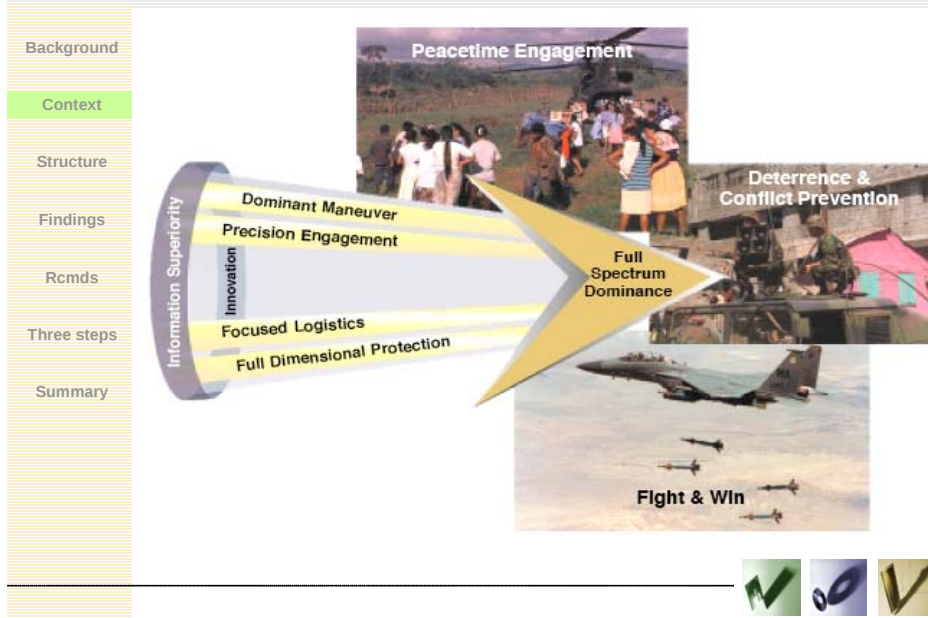
Other Government

- NAVAIR Software Engineering--Tony Guido, NAVAIR SSC
- Model Driven Architecture--Mr. Noel Longuemare
- World Class Modeling Initiatives--Ms. Sunny Conwell, N81
- Software Producability Initiative--Rob Gold, DDRE
- DASN IWS--Dr. Wayne Meeks, Executive Director, Program Executive Office, Integrated Warfare Systems
- Software Process Improvement Initiatives--Mr. Carl Siel, DASN CHENG
- Quadrennial Defense Review (QDR) – Command & Control--CAPT Gary Stark
- Global Information Grid (GIG)/NII--Ms. Priscilla Guthrie, NII
- Army Enterprise Information Technology--Honorable Claude Bolton, Assistant Secretary of the Army (Acquisition, Logistics and Technology)
- National Security Agency (NSA)--Mr. Rob Case, MITRE
- DDR&E Software Research Plans--Rob Gold, DDR&E, Associate Director for Software and Embedded Systems
- Use of Model Driven Architecture (MDA) at GSA--George Thomas, GSA OCIO IAA Chief Architect
- Dr. Michael McDonald, Sandia National Labs
- Defense Science Board (DSB) 2006 Summer Study on Information Management for Net-Centric Operations--LTC Scott Dolgof
- Acquisition Center of Excellence (ACE) History--Page Glennie, DASN C4I
- Defense Modeling & Simulation Office (DMSO)--Capt. Mike Lilientahl
- Software Technology Service Center (STSC)--Dan Bennett

This page intentionally left blank



Joint Vision 2020



Joint Vision 2020

The importance of information dominance is widely recognized, as suggested by this graphic from the Joint Vision 2020 policy statement. Of course the ability to build large software systems, with all the usual properties, is an obvious prerequisite to information dominance. Thus, software problems directly block the forward movement of a central component of defense strategy in the 21st century.

This page intentionally left blank



More capability and lower cost

Background

Context

Structure

Findings

Rcmds

Three steps

Summary

- Software enables new capabilities, such as:
 - Information gathering, fusion, and distribution
 - Coalition collaboration
 - Intelligence gathering
- Software advantages relative to hardware
 - Zero cost replication
 - Greater flexibility
 - Easier upgrade
 - Superior SWAP (Size, Weight, and Power)



More capability and lower cost

Why is software so important and central to our strategic vision?

The obvious reason is that software is the great enabler, making possible operations that otherwise would be inconceivable.

The subtle reason software is important is that software often can do what hardware can do, in principle, but software can do it at much lower cost and with other superior qualities. Examples include the growing use of software in synthetic aperture radar and multi-function antennas.

Another much talked about example is the idea of a software radio, capable of providing functions that otherwise would have to be realized with a cartload of capacitors, inductors, and other electronic paraphernalia. With this context, accordingly, JTRS (Joint Tactical Radio System), a program to replace the hardware-intensive radios currently in use with software-based radios, should be a great success. However, somehow the JTRS program has become the poster child for software headaches. In a sense, the JTRS problems, in the face of the promise of the software radio idea, are testimony to the need for this study.

This page intentionally left blank



The playing field

Background

Context

Structure

Findings

Rcmds

Three steps

Summary

- “...the continued development and proliferation of **information technologies will substantially change the conduct of military operations**. These changes in the information environment make *information superiority* a key enabler of the transformation of the operational capabilities of the joint force and the evolution of joint command and control... **Information superiority is the critical enabler of the transformation** of the Department ...”

From Joint Vision 2020
General Henry Shelton, CJCS, 2000

- “Key to achieving this full spectrum dominance will be the ability of U.S. forces to acquire information superiority and the technologies that enable it.”

Dolores Etter, DDR&E, DUSDA&T, 2000



The playing field

For at least a thousand years, great warfare strategists were adherents of the notion of “know thy enemy.” In more modern times, strategic and battlefield intelligence gathering and dissemination were keys to successful battles and campaigns. Only in the digital age, however, has the notion of information superiority been understood as the critical enabler for full spectrum dominance.

In the early 1990s, the idea of information superiority appear in numerous student papers at the Naval War College. By 1997 Joint Warfare Science and Technology Plan, DoD and the Joint Staff called for the goal of information superiority to enable rapid conflict resolution.

The CJCS (Chairman of the Joint Chiefs of Staff) Joint Vision 2020 focused attention on full spectrum dominance – achieved through the “interdependent application of dominant maneuver, precision engagement, focused logistics, and full dimensional protection. Attaining that goal requires the steady infusion of new technology and modernization and replacement of equipment. However, material superiority alone is not sufficient. Of greater importance is the development of doctrine, organizations, training and education, leaders, and people that effectively take advantage of the technology.”

The evolution of these elements over the next two decades will be strongly influenced by the continued development and proliferation of information technologies, and will substantially change the conduct of military operations. These changes in the information environment make information superiority a key enabler of the transformation of the operational capabilities of the joint force and the evolution of joint command and control.

Testimony to the importance of information dominance is easy to find. The quotes above are representative. Note, however, that both come from the heady and optimistic time,

near the peak in the internet boom, when the United States was the undisputed leader in information technology. Today, the United States has slipped in many areas it previously dominated.



Human resources

The pipeline is running dry

Background

Context

Structure

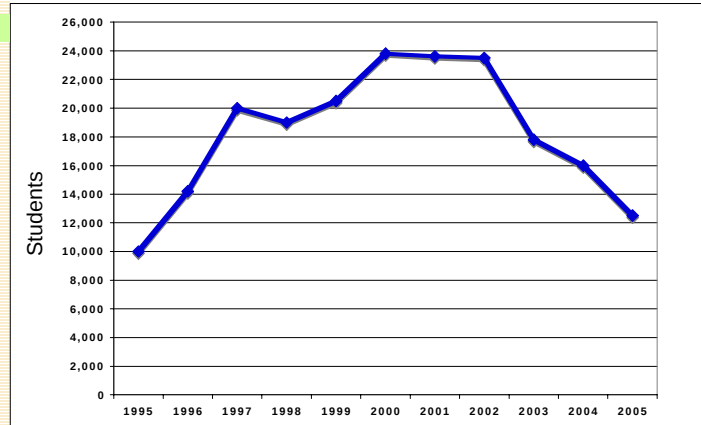
Findings

Rcmds

Three steps

Summary

US CS/CE Undergraduate Majors



May 2006 Computing Research News

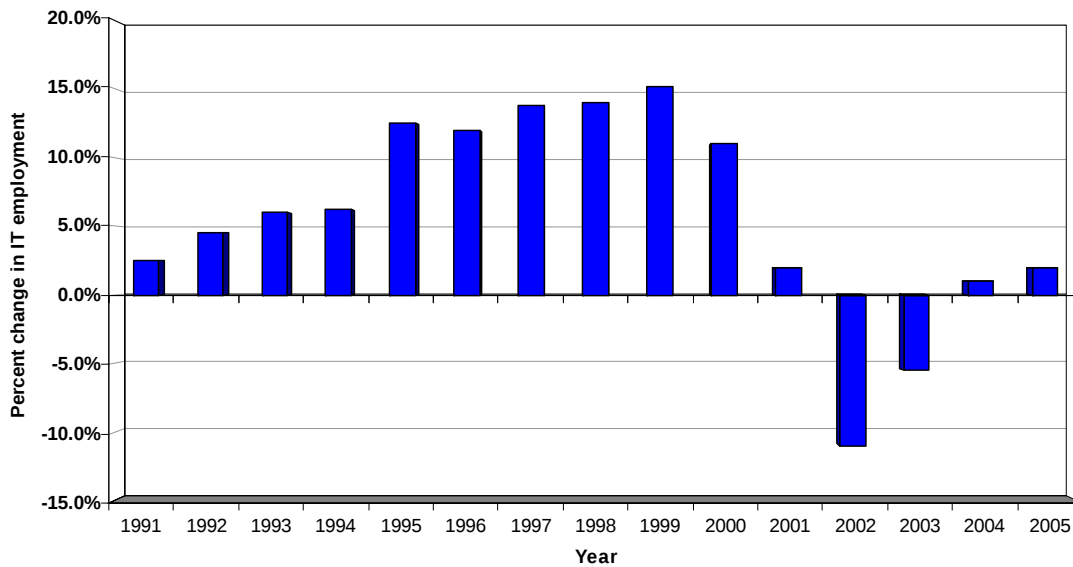


Human resources - The pipeline is running dry

Every year, the Computing Research Association conducts the Taulbee Survey which analyzes the enrollment, production, and employment of computer science students and graduates. This survey has been performed annually since 1974. Recent reports are available on-line at <http://www.cra.org/statistics/>

According to the survey, enrollment in computer science/software engineering courses of study has been on the average decreasing since the mid-1980s. While the number of Bachelors degrees conferred had risen slightly between 1996 and 2000, the number of degrees experienced a 13% decrease in academic year 2004-2005 when compared to the previous year. In addition, the number of newly declared CS/CE undergraduates has been decreasing since 2000, suggesting that the supply of fresh CS/CE graduates will continue to decline. In the mid-1980s, approximately 5% of incoming freshmen chose CS/CE as their declared major. This level dropped in the early 1990s to approximately 1.5%, then rose to about 3.5% in 2000. Currently, the proportion is below 1.5% and steadily declining. [Source- Computing Research News, May 2006]

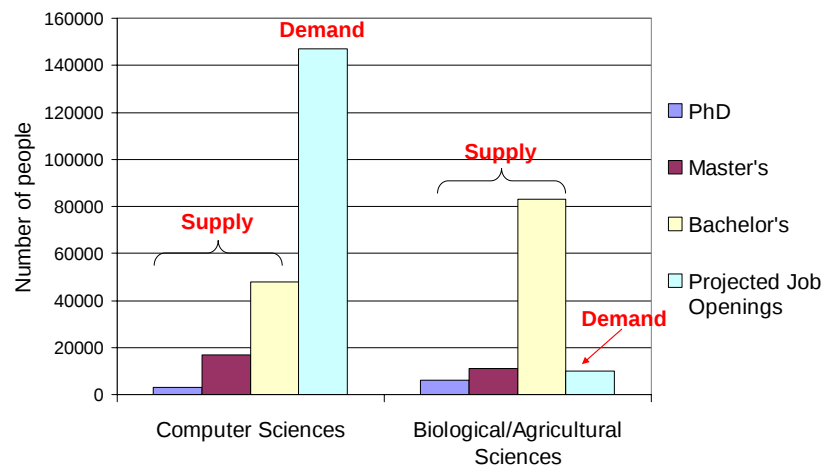
One possible event that may have contributed to this decline is the reduction in IT jobs that occurred after the 2001 recession when, between 2001 and 2004, the number of IT jobs shrank from about 2.1 million to just over 1.7 million. This trend has reversed since 2004, with the job growth currently at about 2%. This is in contrast, however, to 1999 when the job growth was around 15%.



Percent change in IT employment between 1991 and 2005

With the increasing dependence on software-based systems and their growing complexity and size, future needs for trained staff will increase significantly. However, the shrinking pool of trained talent will place any plans for future system development in jeopardy.

In contrast to the shrinking supply, future demands for computing professionals are expected to be significant, with almost 150,000 openings presently available. However, with the current population of computer professionals and the shrinking enrollments at universities, the supply of labor to satisfy this demand will fall short by nearly 90,000 people. In contrast, the supply of professionals in the biological/agricultural sciences is expected to be at 100,000, while the openings in this field will be at about 10,000, resulting in many graduates changing careers. (source – Computing Research Association http://www.cra.org/govaffairs/blog/projected_job_openings.pdf)



The implication is that the ability to hire sufficient professional staff to develop future systems will be a significant challenge. Candidates will likely select jobs based on salary and attractiveness of the position. DoD programs will have to compete with non-DoD business entities, such as web design, corporate information systems, and other commercial activities. The effect will be felt in the ability to develop new systems. If staff can be hired at all, the salaries for this staff may be significantly higher than what is being paid currently. The risk is that the standards for such staff may be lowered simply to hire more people at salaries in line with past experience.

To mitigate such risks, there are few options available. One is to increase the productivity of staff to achieve a multiplicative effect. Such productivity increases could be achieved through the use of automated tools, transferring the effort from human to machine, and allowing the human to focus on the more complex aspects of the system development. An ancillary benefit is that by relying on tools to perform many of the stages of development, the likelihood of defect introduction will be reduced.

Another option is to recognize that not all software engineers are created equal. It has been observed that the range of capability among engineers can be as large as 100-1 [Source: "Not all Programmers Are Created Equal," G. Edward Bryan, IEEE, 1994]. By focusing on hiring the most capable in this range, the effect will be to achieve dramatic increases in productivity. However, as the demand for such talent increases, their cost is likely to rise dramatically. The current approach used by companies when hiring IT staff is to narrowly focus on the specific skills needed for the job. Whether this approach is successful at identifying the most capable of programmers is debatable. In particular, whether this approach is able to identify those software developers whose training allows them to continue to grow rather than being locked into specific, end-of-life technologies, is not clear.

This page intentionally left blank



Globalizing of Software and Hardware

Background

Context

Structure

Findings

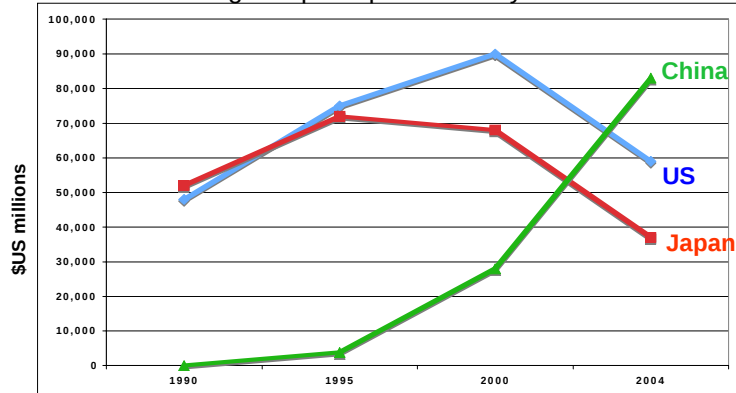
Recoms

Three steps

Summary

- 470,000 IT jobs outsourced overseas, ~25%
- 80% of 300mm fabrication factories are overseas

Leading computer producers by location



Source: Reed Electronics Research, Yearbook of World Electronics Data



Globalizing of Software and Hardware

In order to fulfill the growing needs, companies have been exploiting global resources, outsourcing their IT jobs to other countries such as India, Russia, and China, as well as making use of the limited number of H-1B visas that have been made available. With offshoring, not only are the labor rates lower, but in many of these countries, there are ample supplies of well-educated, trained people. It is estimated that approximately 470,000 IT jobs have been offshored. Since there are currently approximately 1.8 million IT-related jobs in the U.S., the proportion of offshored jobs represents about 25% of U.S. employment. [Source: Snigdha Srivastava and Nik Theodore. *Information Technology Labor Markets: Rebounding, But Slowly*. Center for Urban Economic Development, University of Illinois at Chicago. June 2006] Due to security concerns, however, the offshoring option is largely unavailable to DoD programs, except perhaps under tightly controlled conditions. An important side-effect of this trend is that part of the IT experience and talent base is moving off shore.

Note that the accuracy for many of these statistics is not guaranteed. The data collection processes vary among the researchers, and the interpretation of what an IT job is remains inconsistent across the various studies. Nonetheless, the general trend indicated is consistent across studies.

Likewise, for computer manufacturers, the trend towards offshoring has been significant. In a three year period, the proportion of 300mm fabrication plants in the U.S. has decreased from 30% worldwide to 20% [Source: Defense Science Board Task Force Report On High Performance Microchip Supply February 2005]. Additionally, the U.S. is no longer the dominant producer of computers. China has moved from being an insignificant source of

computers to being the largest source in the world, having surpassed the U.S. in 2003. Such a trend increases the dependence of the U.S. on foreign manufacturers.



Spending

Background

Context

Structure

Findings

Rcmds

Three steps

Summary

Fiscal 2005 Defense Budget Proposal, spending by category (amounts in billions)			
	2003	2004	2005
Business applications	\$5.4	\$5.2	\$5.0
Warfighting and national security systems	\$6.4	\$7.0	\$7.8
Shared infrastructure and information assurance activities	\$14.6	\$15.0	\$14.8
Other IT	\$1.0	\$0.9	\$1.1
Total	\$27.3	\$28.2	\$28.7

Fiscal 2005 Defense Budget Proposal, spending by service (amounts in billions)			
	2003	2004	2005
Army	\$6.2	\$5.5	\$5.4
Navy	\$5.6	\$6.6	\$6.6
Air Force	\$5.2	\$5.7	\$6.4
DOD	\$10.3	\$10.4	\$10.3
Total	\$27.3	\$28.2	\$28.8

app. \$21B for NSS
(incl combat systems)



Spending

Since 2003, DoD spending on IT has increased from \$27.3B to \$28.7B, an increase of 5%. The spending on warfighting and NSS, however, has increased by more than 21%. During this same period, DoN spending increased by almost 18% on all IT systems.

DoD spending on software

The following table summarizes the IT-related spending in the Defense Budget for the years 2003 to 2005, as provided by John Stenbit [Source: "Stenbit explains DoD IT spending to lawmakers". Government Computing News. 03/22/04]. The shaded cells represent those expenditures relating to warfighting systems as opposed to pure IT systems.

Fiscal 2005 Defense Budget Proposal, spending by category (amounts in billions)			
	2003	2004	2005
Business applications	\$5.36	\$5.21	\$5.03
Warfighting and national security systems	\$6.38	\$7.01	\$7.78
Shared infrastructure and information assurance activities	\$14.57	\$15.05	\$14.83
Other IT	\$1.01	\$0.96	\$1.08
Total	\$27.33	\$28.24	\$28.72

From 2003 to 2005, DoD spending on IT increased from \$27.3B to \$28.7B, an increase of 5%. The spending on warfighting and National Security Systems (NSS), however, has increased by more than 21%, from \$6.38B to \$7.78B.

During this same period, DoN spending increased by almost 18% on all IT systems.

Fiscal 2005 Defense Budget Proposal, spending by service (amounts in billions)			
	2003	2004	2005
Army	\$6.23	\$5.51	\$5.45
Navy	\$5.59	\$6.63	\$6.59
Air Force	\$5.25	\$5.71	\$6.40
DoD	\$10.27	\$10.40	\$10.28
Total	\$27.33	\$28.24	\$28.72

According to the Government Accountability Office (GAO) and other sources, of the \$21B spent by the DOD on combat systems (including infrastructure investments as well as NSS), as much as 40% is estimated to be attributable to rework costs, defined as effort consumed by fixing problems found in the system during development and test.

If we estimate the Navy's spending for NSS to be approximately \$4.3B, the cost of rework to the Navy would be about \$1.7B. This amount of rework represents a significant impact on the ability to field necessary capabilities, since the funds are diverted from forward engineering to fixing latent problems in the systems.



Impact of rework costs (FY2003)

Background

Context

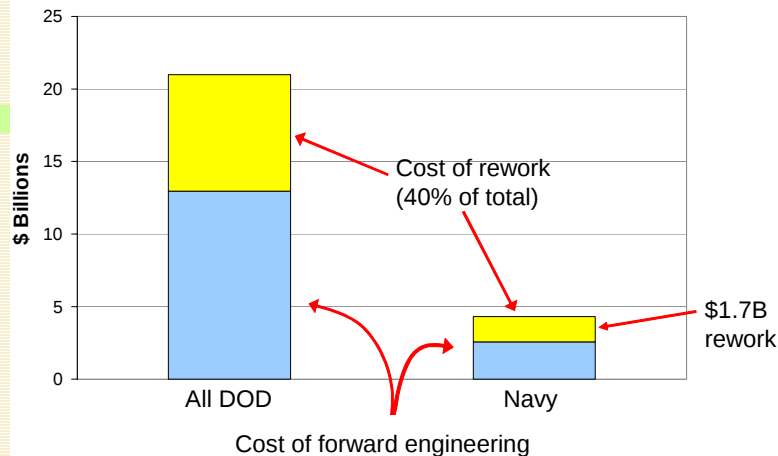
Structure

Findings

Rcmds

Three steps

Summary



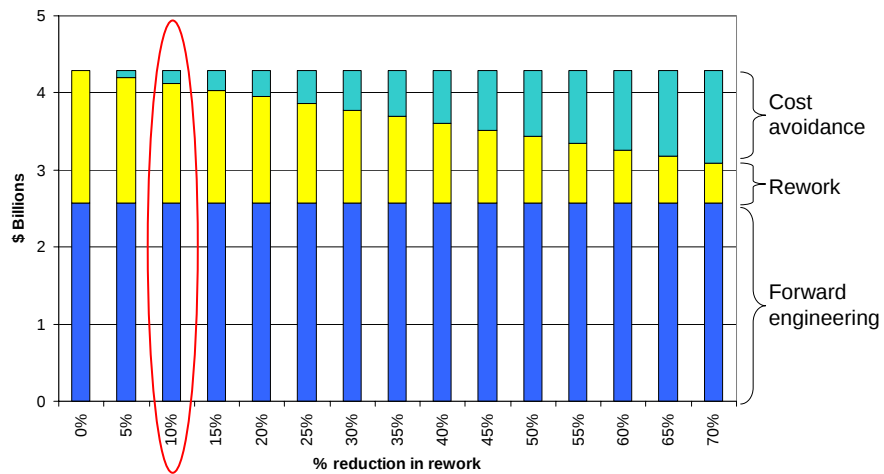
Impact of rework costs (FY2003)

With the 40% estimate for rework costs, of the \$21B spent by the DoD on combat systems (including infrastructure investments as well as NSS), the actual investment effect was equivalent to app. \$13B, the rest being consumed by fixing problems found during development and test.

For the Navy, with the spending having been about \$4.3B, the actual investment totaled approximately \$2.6B, with the remaining \$1.7B being attributed to rework.

If the amount of rework could be reduced, the primary result will be a cost avoidance, allowing the development of additional capabilities for the same overall investment. A secondary but important effect would also be a reduction in the need for increasing numbers of IT staff, thereby avoiding the challenge of finding sufficient numbers of computer-literate candidates.

For the DoN, a reduction of rework on the order of 10% (a modest goal) should result in a cost avoidance of about \$170M. Note that it is not likely that the proportion of rework will ever be reduced to zero – with software development being primarily an R&D activity, it is not realistic to expect that the introduction of defects will be eliminated. However, with the improvement of development processes, experience has shown that a significant reduction in defects can be realized [source: <http://www.thedacs.com>].



A reduction of 10% of rework will result in a cost avoidance of \$170,000,000



Size of typical Naval combat systems

Background

Context

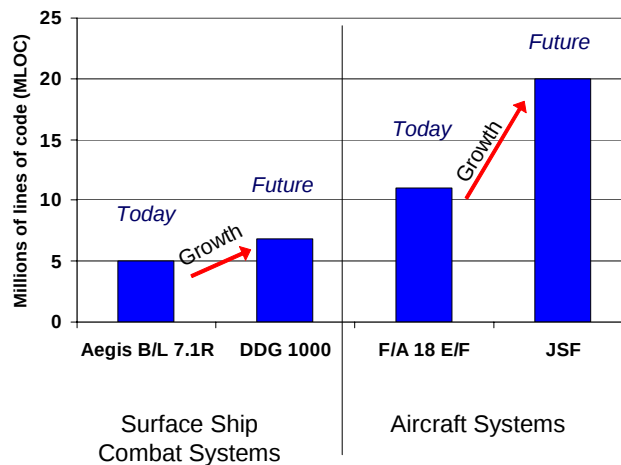
Structure

Findings

Rcmds

Three steps

Summary



System size

The reliance of the DoD on software to provide functionality in our systems has increased significantly. For example, the size of the DDG 1000 combat system is expected to be almost 1.8 MSLOC larger than Aegis Baseline 7.1R, a 36% increase. This growth parallels commercial industry where a similar trend has been observed. This growth is expected - due to its nature, software has enabled us to achieve levels of capability and performance previously unattainable, and perhaps impossible to realize in hardware. However, as the systems get larger and as they become more complex, the challenge of developing them grows significantly. Not unexpectedly, corresponding with the growth, we have seen an increase in cost and schedule overruns.

This growth has resulted in an increased demand for skilled software professionals as well as software system engineers that can decompose complex systems and defined requirements. As systems become more and more complex, this demand will continue to increase, for both DOD and industry sectors.

This page intentionally left blank



History of study

Background

Context

Structure

Findings

Rcmds

Three steps

Summary

- DSB Task Force on Military Software (1987):
“Many previous studies have provided an abundance of valid conclusions and detailed recommendations. Most remain unimplemented.”
- DSB Task Force on Defense Software (2000):
“The Task Force reviewed six major DoD-wide studies that had been performed on software development and acquisition since 1987. These studies contained 134 recommendations, of which only a very few have been implemented.”

Is anybody listening?



History of study

Reports of the Defense Science Board over a twenty year period identify software problems as a favorite topic of study. These reports note that the studies produced have had little effect, however. Of the many recommendations produced (on average about twenty-two per study), only one, recommending the creation of the Software Engineering Institute, has had a significant lasting effect. Two others were the creation of the Ada programming language and the STARS program (Software Technology for Adaptive, Reliable Systems). Beyond these three, recommendations which generated specific, significant actions are hard to identify.

We surmise that the lack of response derives, at least in part, from the lack of DoD organization(s) with a size, mission, competence, and authority to act on software issues. This lack of action, we believe, is symptomatic of a problem not just with recommendations, but with software development and production in general.

This page intentionally left blank



Our central recommendation: structural innovation

Background

Context

Structure

Findings

Rcmds

Three steps

Summary

1. **Mobilize** in the short term:
Rapid Evolution Software Engineering Teams (RESET)
2. **Transform** in the midterm:
A Naval Software System Center
3. **Consolidate** in the long term:
 - Status quo after step two?
 - A Naval warfare center?
 - A joint warfare center?



Our central recommendation: structural innovation

In the course of our study, we identified many problems and we suggest corresponding solutions. Implementing these solutions is unlikely, however, without some structural innovation because there is no DoN-wide organization with the mission and authority to evaluate our suggestions and follow through on implementation.

In light of the importance of software to our defense in the 21st century, we believe that there is a need to create a DoN-wide organization with a software mission and appropriate authority. In today's climate, however, the creation of such an organization has to proceed in steps. Therefore, we have developed a three-step plan.

Our plan is to start small, with approximately 40 people deployed in what we call RESET (Rapid Evolution of Software Engineering Technology) teams. The idea is to concentrate forces, augment selected programs with talented people, extend the program management function, and generally demonstrate that emerging technology, coupled with intelligent acquisition policy, can accomplish great cost savings and quality improvements. We are confident that such teams can be staffed and made effective because we have seen activities in the Army's Future Combat System (FCS) program and in NAVAIR that have some elements in common with our RESET team idea.

Once the RESET teams demonstrate success, the next step involves development of a more ambitious element - the Naval Software Center. The Naval Software Center would add DoN-wide missions to the program-oriented activities of the RESET teams.

Eventually, we see the Navy Software Center evolving into something with considerable size and stature. The exact shape is a matter of debate. There is no particular point in resolving shape at this point, however, without the wisdom to be drawn from

experience gathered in steps one and two. No matter what the final shape, steps one and two seem like logical prerequisites.



Representative Findings

Background

Context

Structure

Findings

Rcmds

Three steps

Summary

- Inadequate system engineering—particularly, requirements definition and system requirements flow-down
- Model driven methods (MDD and MDA) valuable when matched to a task—they are not universal silver bullets
- Few experienced software acquisition professionals
- Programmer productivity varies enormously
- Inadequate application of existing process methodologies
- Inadequate incentives for openness
- Testing, security, and interoperability often too late
- Lack of investment in software engineering research



Representative Findings

Specific findings in representative areas follow:

Program/Defense Industry Findings

The Defense Department spends approximately 40% of its RDT&T budget on software, as cited by the Government Accountability Office (GAO) and the DoD Chief Information Office. In FY 2003, that equated to \$21B (GAO report 2003); and in FY 2006, that amounted to more than \$30B (DOD CIO Report 2006). Of those dollars, approximately 40% was attributed to rework efforts. That amounts to \$8.4B and \$12B respectively. We felt that with that level of rework, there was significant motivation and opportunity for improvement through software tools, process and automation.

In terms of software processes, we noted a lack of adequate system engineering from top to bottom. This included addressing requirements definition, specification flow-down, compliance, plans for risk management and final test verification. There was a lack of system engineering processes and use of productivity enhancing tools which would allow for timely evolutionary acquisition (spiral development).

Programs were often too optimistic about their ability to reuse software from other product areas. Comments within software code was often lacking, making it difficult to understand code functions. Original coders were often not available (reassigned or gone), again leading to problems understanding the code function. Programs also underestimated the requirements and number of lines of code. A result of such problems is often a re-bse-lined program, such as Joint Tactical Radio System re-bse-lining.

One positive attribute which we found was the driving force within the DoD to promote Capability Maturity Model Integration (CMMI) certifications in order to develop

common software practices. However, the CMMI certifications are not perfect. A facility developing a DoD program could be certified to a certain CMMI level; but this did not ensure that the certified processes were appropriate for the system under development, that the software engineers working on the program were part of the original certification, or that the certified processes were actually followed.

Claude Bolton, Assistant Secretary of the Army (Acquisition, Logistics and Technology) told us about the Army's Strategic Software Improvement Program (ASSIP) Plan. ASSIP addresses the sustained efforts necessary to institutionalize continuous improvement in acquisition practices for software intensive systems across the system lifecycle. It also addresses the need to proactively identify, create, and mature the software and system engineering technologies that will be required in response to the increasing complexity of software. We found no analogous effort or champion on the part of the Navy with respect to a similar type of initiative or software focus.

In terms of management, software quality and productivity seem to be heavily influenced by the competency of the leadership and adequacy of management practices. We felt that it was not necessarily the specific process that determined quality or effectiveness, as much as the presence of good leadership and consistent direction. For example, in the Navy's Single Integrated Air Picture Program, Capt. Jeff Wilson and, in the Army's Future Combat System Program, Lt. Col David Basset give such leadership and direction - thus providing the perquisite for success. Instituting good leadership and management practices in software development has a significant potential benefit.

In terms of enabling technology that can supplement good leadership, processes such as Model Driven Development™ (MDD™) and Model Driven Architecture® (MDA®) appear to offer potential for structure and oversight in the development cycle. We heard from briefers who were strong advocates for processes such as MDD™ (for example: the SIAP Program- using Kennedy Carter's iUML tool; Raytheon - using the PathMATE™ tool). These processes use higher level tools and a language (such as UML) to define software as a system of systems and thus assist in generating, evaluating, and formalizing requirements. Higher level tools can also help in testing and validation of system functions and interfaces.

None of these tools is a silver bullet, however. None can be said to be fully mature. None can be applied without careful attention to the principle that the tool should be well matched to the problem. There is no seamless requirements-to-models-to-code-to-test-to-lifecycle solution at all.

Several key authorities noted that people's capability and skill sets were fundamental to successful software development. A classic study indicates that top programmers are as much as 200 times more productive than the poorest performers. The study found that the top 27% of the staff did 78% of the work, yet the overall salary range was a factor of only two. So for example, in the most extreme case, if we have a 200 day project, the best of programmers could do the work in one day, while the weakest of programmers would take 200 days. If we assume that the best programmer is paid twice what the weak programmer receives, using weak programmers cost 100 times more than using strong programmers.

We found that in the Defense Industry, there was incentive to use lower productivity programmers. With Cost-Plus-Fixed-Fee contract structures, there exists an incentive to use

lower skilled labor. Hence, if the task is not completed within the specified labor-hours, more hours can be added to complete the work.

Commercial Findings

During fact-finding, we had a two-day meeting in Redmond with the Microsoft product line development organization and with their advanced research organization. Our purpose was to gain insight into the strategies, tools and procedures used for software development in Microsoft's Windows OS product line, as well as for other application software development. One of our principal findings was that Microsoft did not mandate particular software specification and development tools to be used by developers. This was primarily true for the new Windows VISTA OS development for which the size of the total software system is approximately 50 million lines of code. Their strong belief was that these types of tools are typically very domain-specific, and therefore they rely on each development team to select and develop tools that best fit their particular functional requirement. As an example, in the VISTA Windows OS development, they do not use model driven architecture or model driven design technologies. However, Microsoft is a very strong believer in the incorporation of integrated, centralized testing as a part of their developmental software system build process to be used throughout development. To help facilitate testing for both errors and functionality validation, Microsoft has developed and now uses a software annotation language toolset called SAL as part of their developmental process. In addition, as soon as possible in the development process, they also institute a regular, time-based software system build process and submit every new build to thorough testing before committing any software elements to the new system build. New system builds of the VISTA OS were occurring each day. At the appropriate time in the development process, it is required that all developers actually use the latest build of the developmental system as their base for doing further software development. This process has been thoroughly incorporated into the Microsoft culture for their major software product lines like Windows VISTA.

Although most of the Windows VISTA OS is written in C++, Microsoft's most recent experience has been that advances in compilers, context-sensitive garbage collection and other technologies are now beginning to make "safe" languages such as JAVA and C# practical for production applications requiring low latency, efficient resource utilization and predictable performance. There is some evidence that these "safe" languages could become the future for Commercial Off-the-Shelf (COTS) software. In addition to these developments, Microsoft Research is also in the process of developing next generation versions of Window OS, which use layering and internal inter-process access controls to achieve much higher degrees of security protection than do current systems. These are probably five or more years away from production. It was also observed at Microsoft, and is recognized to be true for most other vendors, that commercial industry is very heavily focused on the implementation of software systems which are compatible with service oriented architectures (SOA). This is of course completely driven by commercial business' use of the Internet and Web-enabled services. A large proportion of this technology is completely transferable to help meet DoN's operational requirements for systems based on service-oriented-architectures using the GIG and Maritime Framework for the GIG (ForceNet). Commercial developments in the areas of information discovery and distributed collaboration toolsets are of particular interest. We also found that distributed software

systems based on SOA are becoming capable of supporting low latency, time-critical applications which often characterize operational requirements of DoN systems such as the Cooperative Engagement Capability (CEC) system. This is becoming more feasible through advances in commercial Internet communications technologies. These enable the negotiation of low latency, variable quality of service protocols with variable levels of security service functionality.

DoN Findings

The Navy often develops Naval Software in programmatic “boxes” or in organizational “stovepipes” without proper consideration for interoperability between and among programs. As a result, opportunities for software commonality and related cost savings are frequently missed. In addition, currently the DoN has no process to look across platforms and programs to identify such opportunities for portability and reuse.

Development, and more importantly, implementation of a formal interoperability process would identify opportunities for commonality and reuse. On those few occasions when interoperability is considered in the construction of Naval Software, it is most often an afterthought, rather than being designed into the system architecture – a necessity for the development of efficient, effective, reusable interoperable code. This same failure applies to software security. Software is not secure unless the security is architected into the computer system design at the beginning. Neither interoperability nor security are effective “add-ons.”

The Navy not only misses interoperability opportunities, but also often develops software on an ad hoc (i.e., “one off”) basis, without the benefit of applying lessons learned from predecessor programs, and often fails to incorporate commercial best practices. In short, each time the Navy develops software, they “reinvent the wheel.”

Another deficiency we noted was the lack of DoN investment in software technology research. The Office of Naval Research does not have a specific focal area for software research; software development is addressed on a need/specific program basis only. There is no transcending software research group.

We feel that it is a fatal flaw for the Navy to rely solely on the Defense Acquisition Workforce Improvement Act (DAWIA) to provide Navy program managers with the necessary knowledge, skills and abilities. DAWIA requirements for program management level III certification do not account for or include the specialized knowledge, skills and experience for the acquisition of software intensive systems (specifically knowledge and experience in the development and implementation of software) - a particularly high risk because all warfare systems, regardless of size, have a substantially software content.

The lack of software development experience is found throughout all levels of the acquisition hierarchy. As a result, legal and quality issues continue to come to the forefront and Naval contracts lack the incentives for industry to hire quality developers. Naval contractors are further hampered in recruiting by citizenship and security clearance issues.

Standards and metrics are also lacking in the development and acquisition of systems. Programmer skill levels, whether in industry or the government, are often inadequate, and simply relying on CMMI certification does not guarantee that processes will be appropriately selected and followed or that trained personnel will execute. An additional risk is that CMMI certification is done only once and never requires an audit.

For software management, acquisition and quality control there is no focal point to ensure consistency, efficiency, and effectiveness. Finally, only scant attention is paid to requirements management, testing and maintenance costs when contracts are awarded.

This page intentionally left blank



Leadership recommendations

Background

Context

Structure

Findings

Rcmds

Three steps

Summary

- Put somebody in charge:
 - Establish acquisition educational standards
 - Promote basic process improvements
 - Increase awareness of software problems, technology, and opportunities
- The ASN (RDA) is already engaged (memo of 15 May 2006)



Leadership recommendations

We feel that to adequately address the deficiencies cited in the findings and to avoid the pitfalls of previous DoD software panels, a central focal-point and advocate is required in the DoN. The ASN is currently acting in this type of role, as demonstrated by her memorandum of 15 May 2006 (Appendix B), which addresses process improvements for software development.

This page intentionally left blank



Acquisition and practice recommendations

Background

Context

Structure

Findings

Rcmds

Three steps

Summary

- Create software acquisition specialty within the Navy
- Develop real incentives to share specifications, interfaces, models, and software (eg ARCI program)
- Apply emerging software engineering tools to appropriate problems
- Deploy system engineering methods that enable specification, implementation, and testing to evolve together



Acquisition and practice recommendations

As previously noted, it is a fatal flaw for the Navy to rely solely on the Defense Acquisition Workforce Improvement Act to provide Navy program managers with the necessary knowledge, skills and abilities to manage software programs. A lack of software development experience is found throughout all levels of the acquisition hierarchy. As a result, legal and quality issues continue to come to the forefront and Naval contracts lack the incentives for industry to hire quality developers. Naval contractors are further hampered in recruiting by citizenship and security clearance issues.

Other recommendations are offered for data rights, human resources, programming methodologies and security and survivability.

Data Rights

A contentious issue related to software is data rights. In some instances, ownership of software may be important to the contractor. In almost all cases, however, it is critical to the government to have open architectures and the ability to integrate new software into legacy software. These two desires often create conflicts which lead to legal battles. This issue needs to be addressed appropriately in contract specifications flow (government to contractor and contractor to contractor). Appropriate contract language combined with appropriate incentives need to be used to ensure specifications are shared when appropriate, open architecture is maintained, and opportunities for future government use/modification are maintained throughout software development. We recognize, however, that data rights policy flexibility has been shown to be valuable and continues to be needed in connection with SBIR contracts, because without data rights, small companies would have much less incentive to innovate.

Human Resources

The acquisition and development of software is critically dependent on the people involved in the processes. Software acquisition is so sufficiently different from hardware acquisition that a designation of a software acquisition specialist needs to be established. A person with this designation will have met appropriate training and certification requirements. The software acquisition specialists will also need well trained in-house software specialists to provide technical guidance for the acquisition process. These in-house software specialists must either be acquired or trained. On the contractor side, an issue of concern is the use of CMMI level business-unit certification. When the certification is acquired, it relates only to a specific business-unit. There is no guarantee that the certification level of a business unit actually applies to a team assembled for a specific project. In addition, it appears that some contracting approaches, such as cost plus fixed fee, might encourage contractors to use low performing programmers. Because the performance ratio between high performing programmers and low performing programmers is on the order of 10:1, while the pay difference ratio is closer to 2:1, cost plus fixed fee actually increases the funding to the contractor when low performing programmers are used. This concern should be avoided by applying the appropriate contracting language to ensure that high quality programmers will be used by the contractor. Appropriate auditing should then be applied to ensure the use of high performing programmers.

Programming Methodologies

Navy Software Intensive Systems are extremely complex, often including many millions of Source Lines of Code. These systems are deployed over a long time period in an environment characterized by volatile geo-politics and rapidly changing technology. Navy Software Intensive Systems must be rapidly adaptable to the changing missions that emerge from this environment and they must remain safe, reliable and secure as they are changed. The fact that so much of the functionality is implemented in software is the single most important factor in enabling these systems to evolve to meet changing needs. The Navy therefore needs to develop and adopt methodologies that regard evolvability as the most important characteristic of software systems and that are geared towards supporting a continuous process of improvement while maintaining performance, safety and security. Along these lines, we recommend the following:

Recommendation: Use system engineering methodologies that enable specification, implementation, and testing to evolve together.

The Navy needs to develop an integrated evolutionary Systems Engineering approach that unites specification, implementation, testing and verification into a single continuous process in which all components (specification, design, implementation, testing and verification) evolve together in small steps. This approach grounds the requirements in the reality of what can be implemented and through early modeling tests the requirements against what the users actually need. It helps to keep requirements, designs, and implementations in sync with one another and provides a chain of accountability and visibility throughout the process.

Recommendation: Encourage practice of nightly/weekly end-to-end system build and test.

Delaying integration and testing until the end of a development process leads to large costs and delays. We recommend that the Navy adopt a process that is used in commercial software development (e.g. Microsoft) and that involves building a complete system at very frequent intervals (daily or weekly). Each build is thoroughly tested using test suites and verification tools that are co-developed with the software system itself. Systematic check-in processes guarantee that individual components are submitted to the system build only after they have undergone component level testing and verification. By doing this process early and often, defects are detected early and fixed before they have serious consequences. Because there is always a “current system” that has been tested, it becomes possible to issue frequent releases that introduce new capabilities in small blocks and to rapidly fix problems found in the field. Because the testing and verification tools progress with the software, they provide more substantive support and better testing of what actually does go wrong, not just what might go wrong.

Recommendation: Learn to match emerging methodologies and tools to the specific problems at hand.

There are several emerging technologies that will be important to the Navy: Model-driven approaches (e.g. model-driven architecture, model-driven design, model-integrated computing), software factories, product-line approaches, use of annotations and static checking, model-checking, Incremental implementation methodologies using frequent (daily or weekly) full system builds and tests, use of smaller but much more frequent (e.g. quarterly) releases to the field, domain-specific languages, safe languages, configuration management tools, and so on. None of these is a “silver bullet” that fixes all problems; however, each of them has merit. In most cases, more than one of these methodologies and tools may be applied synergistically.

We recommend that the Navy begin to evaluate and adopt these emerging tools and methodologies; however, it is important that the Navy develop sophisticated taste. Some tools and approaches match certain problems far better than others. The Navy should learn to match emerging methodologies and tools to their problems, adopt an experimental attitude, and become willing to try new approaches and tools - adopting them when they fit well and discarding them when they do not. Time should be budgeted into projects to allow for such experimentation and there should be a process for sharing lessons learned.

Recommendation: Avoid proprietary development environments that block migration to competing and future development environments.

While we recommend that the Navy experiment and adopt new tools, we also recommend that the Navy be wary of “lock-in”, whether intentional or accidental. Many of the emerging tools have proprietary elements that provide no path for migration to other tools. It is critical that the Navy avoid such lock-in for several reasons: first, many of the tools are developed by organizations that might go out of business or abandon the product. Second, because no single tool covers all of the issues of interest, it is critical to federate the tools into larger ensembles. Third, newer tools with better capabilities are always emerging. Therefore, we recommend that the Navy not commit any project to a tool that does not offer migration paths to other environments.

Security and Survivability

Navy Software Intensive Systems will operate as components of an overall Net-centric military, integrated through ForceNet and the GIG. Traditionally, most of the Navy's Software Intensive Systems have been protected by an "air-gap"; but we are rapidly moving into a world in which all systems will be networked, including those, such as weapon systems, that traditionally operated stand-alone. With the power of net-centric warfare, however, also comes the vulnerability to attacks on the information systems that manage sensors, fire-control, and C². These attacks may steal or compromise information; they may disable or degrade entire systems. As we have seen in the commercial sector, vulnerabilities in networked software systems can be rapidly amplified.

Navy software systems must be designed systematically for security and survivability. That is to say they must be designed both to resist attacks and to be able to provide critical services even in the face of successful attacks. These concerns need to be addressed as a critical element of the overall development process: systematic vulnerability analyses should be conducted early in the requirements and specification process; security and survivability approaches need to be included as a key element of the design process; the design solutions need to be documented, modeled and analyzed; the implementations need to be systematically checked for compliance with the security design; and the overall system needs to be tested and validated not just for functionality but also for security and survivability.

Recommendation: Require security and survivability specifications.

All software intensive systems will operate in a networked environment; information attacks will be an integral part of the war-fighting environment. Every software intensive system will have security and survivability requirements, and therefore, every such system must include specifications that address these requirements. These should be rooted in a systematic vulnerability analysis that documents the threat environment. Automated vulnerability analysis tools, where available, should be employed in tandem with more traditional manual analysis techniques.

Recommendation: Require continuous testing and verification for security and survivability properties.

As the system design and implementation proceed, it is crucial that security and survivability specifications are adhered to. Thus, test suites that stress these properties should be developed early. These should be exercised constantly and used in conjunction with verification tools to ascertain whether each system build meets the security and survivability design goals. Independent review team efforts should be conducted periodically to provide independent certification.

Recommendation: Track and exploit emerging systems definition models to manage security and survivability.

The commercial sector is developing a set of system definition and modeling languages that are used to drive the system configuration process. These tools can help guarantee that each component of an overall distributed system adheres to a set of configuration constraints that have security implications. These tools are not yet mature, but

there are emerging standards and tools. The Navy should track this emerging technology and require its use when there is a good fit.

Recommendation: Require use of safe languages to remove vulnerabilities.

It is estimated that 80% of the security vulnerabilities in software intensive systems arise from “buffer overflows” - the failure of a program to check that it does not overrun the bounds of an array when storing data. Some programming languages automatically perform bounds checks and other similar checks, and are therefore referred to as safe languages - these include Java, Lisp, C#, Python and many others. Other languages, such as C and C++, regard the run-time performance cost of these tests to be too high and would rather leave it to the programmer to manually insert such checks where they think it is appropriate. In practice, if these checks are left to the programmer’s discretion, they are omitted, leading to vulnerabilities.

We have seen convincing evidence that safe languages can perform at the same performance levels as unsafe languages (and sometimes better) and that they offer much higher productivity. We therefore believe that there is little justification for the continued ubiquitous use of unsafe languages in systems programming. We do not believe that it is useful to mandate a particular safe programming language; different ones have different strengths. However, we do recommend that the Navy require the use of safe languages in all new projects.

This page intentionally left blank



Recommendation focus: the user-requirements loop

Background

Context

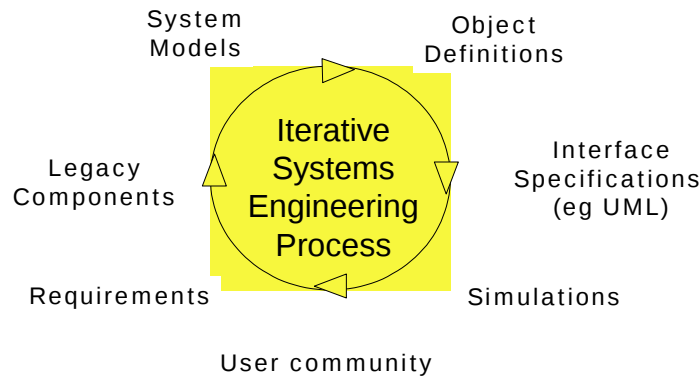
Structure

Findings

Rcmds

Three steps

Summary



**Model driven tools can stimulate and enforce
iterative systems engineering**



Recommendation focus: the user requirements loop

In an ideal world, the connection of the user community back to itself through requirements, models, and simulations would be a tightly coupled loop that would ensure that users expectations are in conformance with reality and that builders produce what users need. Today, promising new tools are emerging that support parts of the loop, such as widely advocated object-definition tools, but collectively, the tools lie in disconnected clumps. Consequently, the output of a requirements system may be ignored by busy downstream developers who have no mechanism for feeding back practical considerations into the requirements system. Requirements, models, simulations, and the eventual system fall out of synchrony ensuring user dissatisfaction and maintenance nightmares.

Thus, there is a conspicuous need for software-supported mechanisms that ensure that users and key stakeholders can effectively communicate with systems engineers during the development and technical validation of system requirements and continuously throughout the evolution of the program.

We are encouraged by the development of higher order language-based system modeling tools for characterizing and then simulating performance of complex software systems; they offer a significant potential for helping users to become players along with software systems engineers in the iterative process of developing requirements for complex software systems and then validating those requirements through high level system operational simulations. But the new tools, by themselves, do not close the loop and cannot ensure that unrealistic or unachievable system requirements can be exposed and reviewed early in the program development process. Similarly, assumptions about interoperation of new software subsystems with existing legacy software systems cannot be tested at a high level before large software developments are undertaken. Also, the validity of assumptions

about the reuse and integration of previously developed software subsystem modules cannot be validated before non-recoverable costs are expended.

Overall, there is a great need for system level tools that could help well-intentioned people to do a good job of understanding and iterating requirements with users as necessary to facilitate tradeoffs for developing solid system specifications. To emphasize this point, we found that significant amounts of the software rework problems now being experienced on many Navy development programs actually have roots in requirements/specifications shortfalls occurring early in the program. In particular, as one example, we found several cases where overly optimistic expectations for the reuse of previously developed software for meeting new program requirements were made without adequate means for validating these expectations. Unfortunately, today many of these types of problems are really not visible until late in the program when higher level software system integration and testing finally occurs. Of course, at these later stages in the program development, any corrections can become very extensive with impacts on many other functions thereby resulting in significant cost and delay.



Naval S&T program recommendations

Background

Context

Structure

Findings

Rcmds

Three steps

Summary

- Start focused effort
- Leverage existing software engineering research and practice
- Develop, for example:
 - Software tools for evolutionary systems engineering
 - Practices for automated daily build, test, and evaluation
 - Domain-specific model languages
 - Technology for dealing with legacy systems
 - Means to exploit lessons-learned and best-practices knowledge bases (such as those of NASA, DOE, FAA, and ONR activity at University of Maryland)



Naval S&T program recommendations

We believe the Government is not investing adequately in Software Engineering R&D. The only major investment is at the Software Engineering Institute (SEI). DARPA and ONR no longer play a significant role. Neither DARPA nor ONR have a single program whose principal topic is software engineering; what little investment exists occurs as an ancillary activity within programs with other purposes. Even making extremely liberal assumptions, it appears that the total DoD investment in Software Engineering R&D accounts for less than 1% of the DARPA budget. The commercial sector provides only minor relief: only a few large companies are making substantial R&D investments in software technology - notable among these is Microsoft's recent interest in software quality. However, there is an overall significant shortfall in software research.

Our most important recommendation in this area is that the Navy have an R&D program in software engineering technology. We identified several research foci of interest, but feel the specific focus is less important than first establishing a concentrated software research effort. The Navy program would complement the ongoing industrial research. The following are some of the areas we identified as worthy of investment.

Focus: Evolutionary Systems Engineering

Today's approach to software development divorces requirements definition, system analysis, design, implementation, testing and continuing engineering and maintenance into a pipeline of separate activities with no feedback between them. This leads to requirements creep, unrealistic designs, buggy implementations, cost overruns, schedule slips and inflated life-cycle costs.

Instead, we recommend a new approach that we call evolutionary systems engineering. This approach should integrate the full spectrum of software engineering tasks

into a single, rapid-response process in which each part of the process proceeds in small incremental steps that provide information to both downstream and upstream activities. For example, early modeling and coding efforts might provide information about the true cost of a requirement, while early testing and simulation efforts can catch bugs before they propagate. Another important aspect of this approach is to automatically capture design rationale, analysis, simulation and test results so that we know how the requirements are being met and so that we can examine the impact of an incremental change in any component of engineering chain.

Evolutionary systems engineering will require tools that allow daily/weekly build practices adapted to military systems. These build practices would support incremental testing of all artifacts: specifications, models, and code helping to find and correct problems earlier and lead to more realistic estimates of progress towards goals. These tools should support frequent, incremental releases that rapidly address problems found in the field and that deliver frequent upgrades in capabilities. This rapid cycling of releases will bring experience into the process of prioritizing continuing engineering efforts, with those issues of most importance to the warfighters being addressed first. Also, such evolutionary systems engineering tools will also provide higher confidence in the quality of the delivered products.

Focus: Model Driven Approach – Domain-Specific Modeling and Implementation

Unlike industry software, Navy software must meet safety, security and real-time requirements. Navy software also deals with a broader and different range of subject matter (e.g. radar, sonar, fire-control, avionics, etc.) than does most commercial software. We therefore recommend that part of the task of the R&D program will be to develop tools complementary to those used in industry that can deal with these additional concerns and domains. We recommend that one focus be on the construction of domain-specific modeling frameworks and domain-specific programming languages that facilitate independent expression and analysis of multiple aspects (e.g. security, real-time requirements) of the problem. But it is also imperative that we provide tools for integrating these independent viewpoints and domains into a common framework that supports global analysis of the entire system.

Focus: Improved Safe Languages

Safe languages such as LISP, Java, and C # show great promise for improving the quality, safety and security of software intensive systems; they also promise significant improvements in programmer productivity. In many cases, the abstraction level achievable in such languages is equal to that of the “action semantics” language of model based systems. We recommend that new effort be put into investigating and developing new safe language technology with particular emphasis on meeting the Navy’s needs for real-time, embedded computing, for aspect-oriented decomposition, and for embedding of domain-specific languages.

Focus: Capturing Commonality Across Projects

The Navy builds many systems. Although each of these serves a unique purpose and embeds a unique set of design decisions, there is still enormous commonality among these systems when viewed from an adequate level of abstraction. Ideally, these systems should be built from a common core that is tailored to each specific system by a relatively minor layer of platform-specific software. This idea is referred to as the product line approach. We recommend that the S&T program investigate new tools and techniques that can support the product line approach. In particular, tools that can help track design choices and the reasons for making them will help to separate out product specific aspects of a system from aspects that are of general purpose across the domain.

Focus: Capturing Lessons Learned and Best Practices

The Navy develops a large number of software systems and many of these systems remain in the field for decades undergoing constant evolution and upgrade. These efforts constitute a huge experience base and one that is uniquely tailored to providing information about future large-scale Navy projects. We suggest that another research focus might be on tools that help organizations learn from best practices as well as from mistakes. We recommend that the research address tools which help organizations capture lessons-learned and best-practices and help to efficiently provide this information to those affected when they can best benefit from it.

This page intentionally left blank



Assessment

Background

Context

Structure

Findings

Rcmds

Three steps

Summary

- Information dominance central to defense, but at risk
- Lots of opportunity, but little decisive action to date, for lack of structure
- Visionary action and structural innovation needed



Assessment

There is no doubt that information dominance will be core to future defense efforts, especially for the long war on terrorism. This leads us to serious concerns because we believe that the way the DoD specifies, contracts for, tests, validates, and maintains software is deeply flawed. Furthermore, with budgets escalating from current war efforts, the DoD and DoN can no longer afford to pay the mounting rework costs associated with development programs. In order to address these concerns, we feel that visionary action and structural innovation are needed to ensure immediate and long term results.

This page intentionally left blank



Step one: Rapid Evolution Software Engineering Teams

Background

Context

Structure

Findings

Rcmds

Three steps

Summary

- Staff each with 10-20 full time equivalents
- Complete user-requirements loop
- Promote use of system engineering tools, policies, and practices
- Champion best-practice software methodology emphasizing commonality, evolution, adaptation, reuse, reliability, interoperability, security and rapid response to changing defense needs
- Identify open systems needs and ensure compliance
- Recommend contract incentives
- Monitor progress and sustain support



Step one: Rapid Evolution Software Engineering Teams

The structural change proposed in the study is a three-step process that starts with the creation of a special kind of project management and ends in the establishment of an organization aimed at ensuring excellence in software. We envision that the size and significance of the final organization will be comparable with a warfighting center.

The initial step establishes and uses RESET (Rapid Evolution of Software Engineering Technology) teams to interact with contractor software development teams. The RESET teams, consisting of 10-15 software experts, will be collocated with the contractor software development team and will interact with them on a continuing basis throughout the contract. They will be responsible for working with both the contractor and the user community to ensure that contractor requirements remain consistent with user needs. They will promote the use of system engineering tools, policies, and practices. They will encourage the use of software methodologies which emphasize commonality, evolution, adaptation, security, and rapid response. They will recommend best practices such as regular code builds and tests throughout the development process. As Naval systems move toward ForceNet and the GIG, the RESET team will understand requirements associated with both and ensure that the software development is compliant with them. To encourage contractors to adopt the recommendations of the RESET team, changes in the contract language may be needed. The RESET team will be in position to observe contractor behavior and help recommend an incentive structure that ensures the desired results. Finally, the RESET team will monitor progress, document best practices and lessons learned, and become the knowledge base for software development processes for the government.

This page intentionally left blank



Step one: Implementation

Background

Context

Structure

Findings

Rcmds

Three steps

Summary

- Embed on contractor site in two or more representative programs (to promote commonality), such as CG(X), BAMS, Aegis upgrade, LCS
- ASN RDA provides seed money to selected PEO to initiate activity
- Staff with expert personnel from ONR, NRL, UARC, FFRDC (such as SEI), Warfare Centers, National Laboratories, government agencies, academia, and noncompeting contractors
- Report to ASN through PEO



Step one: Implementation

To best assess the effectiveness of RESET teams, they should initially be embedded in at least two or more representative programs. The programs chosen should be software intensive and the team should be introduced into the program at a point where contracting language can clearly identify the team and its activities. This will avoid any issues associated with non-contracted activities or suggestions from the RESET team. The Navy should consider candidate programs such as CG(X), BAMS, Aegis upgrade and LCS.

The ASN (RDA) will be responsible for providing the funding to establish the team and conduct its activities. The individuals comprising the team should be software experts and should come from agencies such as ONR, NRL, University Affiliated Research Centers (UARC), FFRDCs (such as SEI, IDA, Aerospace, and MITRE), Warfare Centers, National Laboratories, other government agencies, academia, and noncompeting contractors. The teams will report to the ASN through the PEO. This will provide the type of visibility the team will need to influence the program.

This page intentionally left blank



Step two: Naval Software System Center

Background

Context

Structure

Findings

Rcmds

Three steps

Summary

- Staff with ~50 full time equivalents
- Institutionalize and staff RESET teams
- Build models and assist in building models
 - Complete requirements---users loop
 - Complete model---VV&A loop
 - Solve ownership problem
 - Ensure compliance with lessons learned
- Maximize Naval commonality
- Manage and staff independent expert reviews
- Recommend incentives and acquisition policy
- Manage innovation through programs, such as SBIRs, ATDs/JCTDs, ...



Step two: Naval Software System Center

Following the formation and successful operation of two or three RESET teams, the next step is to institutionalize RESET teams and expand the responsibility and benefit of getting the highest quality software system support to the program managers and PEOs. A new structure was envisioned which could provide an enterprise-wide capability in software system development, management support, and lifecycle support. We named this the “Naval Software System Center” (NSSC). This concept has many of the good attributes and functions of the NAVAIR System Software Support Center and the Army Future Combat System Software Product management team. The NSSC has a mission to support and staff the RESET teams; build models and assist in helping elements of the Naval Enterprise in building models; manage innovation; ensure openness and commonality (where it is best suited for interoperability and business case benefits); recommend acquisition policy; ensure users are involved with the software developers defining requirements and manage/staff independent expert reviews.

Comparison of known entities doing similar type of work suggests a staff of approximately 50 full-time equivalents composed of Naval software and system experts, IPAs from FFRDCs, other government agencies, SEI and academia as required.

This page intentionally left blank



Step two: Implementation

Background

Context

Structure

Findings

Rcmds

Three steps

Summary

- Embed in SYSCOM, NRL, or existing warfare center
- ASN RDA funds for FY08 via redirection, then for FY09 as line item
- Report to a PEO, DASN to ASN, and OPNAV
- Enterprise coordination



Step two: Implementation

The NSSC could be embedded in an existing Systems Command, the Naval Research Laboratory (viz. the Artificial Intelligence Center), or an existing Warfare Center.

We envision that funding to stand up and support the NSSC would be required in FY 2000. A budget line item wedge for FY 2009 can be developed over the next year within the normal cycle of budget, build and review process. Funds to support the facility and people are legitimate category 6.5 funds.

The management and reporting line for the NSSC is proposed to be from the PEO (We suggest PEO (IWS) which is cross cutting across all warfare areas) to the ASN (RDA) through a DASN (C4I or IWS) or the CHENG.

This lean organization must be designed to be an enterprise-wide service entity without platform or mission lines and totally open to funding and implementation of the very best, efficient, and secure software technology and practices from all services - inside and outside of the DoD - to include leveraging commercial, other government agencies (e.g., DoE, NASA) and academia best practices.

This page intentionally left blank



Step three: Consolidation

Background

Context

Structure

Findings

Rcmds

Three steps

Summary

- A cross-cutting, horizontally integrated, possibly joint activity that ensures information dominance
- Size and structure to be evolved from experience with steps one and two



Step three: Consolidation

Once the RESET teams and the Naval Software Center have achieved a set of validated processes and structure, the third step would be to establish a permanent, enterprise-wide asset that consolidates practices into naval operations. The objective of this entity would be to continually address acquisition, development, and sustainment of software intensive systems. Moreover, the final vision would be to have this entity perform the cross-cutting, horizontally integrated naval activity that ensures information dominance.

The final consolidated structure should be in equivalent size and significance to a Warfare Center and should report directly to the key naval leadership position such as a PEO. The specific size and structure of the Warfare Center will be determined from the experience and knowledge gained in steps 1 and 2.

This page intentionally left blank



Risks and challenges: steps one–three

Background

Context

Structure

Findings

Rcmds

Three steps

Summary

- Human resources difficult to obtain
- Cultural resistance
- Budget priorities
- Industry pushback
- Contracting difficulties
- Multiyear sustenance



Risks and challenges: steps one - three

There are, of course, numerous risks and challenges associated with successfully implementing the three-step process. There were five areas that we specifically highlight as potential obstacles.

Human resources - The RESET teams, Software Systems Center and final Software Warfare Center will require individuals with a diverse set of skills. The ideal candidates should have a computer science education as well as program acquisition knowledge. These individuals will need to have hands-on code generation experience as well as first-hand software program and procurement management experience. Finding of the right leaders will be a challenge, but we note that officers trained at the Naval Postgraduate School constitute a great pool of candidates. Others could be brought in from institutions outside of the Naval enterprise such as FFRDCs, UARCs, national labs, non-competing contractors, etc.

Cultural resistance - The RESET teams, Naval Software Systems Center, and Software Warfare Center will develop a set of cross-cutting common methodologies, best practices, and standards. There will be different views on preferred approaches among programs and organizations. There may be strong resistance by these groups to changing practices and adapting standard methods. Furthermore, there may be resistance on the part of programs to inserting outside knowledge experts into current operations. (Personalities on the RESET teams could play a crucial role in determining acceptance and success.) The three step process and early successes would help begin to address cultural resistances.

Budget priorities - Initially, funding for the RESET teams and the Software Systems Center will require redirection of budgets and decisions on priorities. Without a strong champion(s) for funding support, the team and center concept cannot be validated. The long-term vision is to sustain the Software Warfare Center through a budget line-item.

Industry pushback - The RESET teams need to work cooperatively with industry and offer guidance, support, information, and validation of user requirements. The structure and approach taken by the RESET teams would help determine their acceptance by industry. Industry would accept these teams if they were viewed as vehicles to help ensure success as opposed to a "tax" to the system.

Contracting difficulties - Special provisions may need to be made to contracts to provide for RESET team interface as well as for implementation of RESET team and Software System Center recommendations.



Summary

Background

Context

Structure

Findings

Rcmds

Three steps

Summary

- Assessed situation and articulated concerns
- Listed findings and recommendations
- Established need for innovative structure
- Identified risks and challenges
- Proposed three-step plan for ASN RDA action

**To maintain information dominance,
inaction is not an option**



Summary

This page intentionally left blank

Appendix A

Terms of Reference

Objective

This study will examine how systems engineering, model driven architecture, and modular software specification and implementation methods can be applied in a realistic manner to specifying, bidding, and engineering of software-intensive systems, across multiple organizations. Beyond technical considerations, this study will investigate challenges with using and fielding computer-based acquisition tools in the context of existing Navy organizational structures, policies, and the overall workflow associated with the acquisition of new systems.

Background

There is a great opportunity to introduce efficiencies, transparencies, and tracking into the overall workflow involved in acquiring new systems and platforms by applying information technology and constructs from information technology in a beginning to end manner. Key S&T is required for understanding how to insert modeling, simulation, and computer-based representations into the design, construction, testing, and maintenance of software-intensive systems. Particular opportunities include the application of ideas developed in industry for the decomposition of software systems into interacting modules with clearly defined interfaces in a comprehensive manner, touching multiple points in the acquisition process.

The study will require the participation of experts with a detailed understanding of the Navy's software systems acquisition process as well as experts on systems engineering, modeling and simulation, and on principles of modular design. The study would build on the prior NRAC study on system modularity, but would direct a focus of attention on the technical and organizational challenges of acquisition.

Specific Taskings

- Review current relevant DoD programs (e.g. Navy Open Architecture, Single Integrated Air Picture, etc.).
- Review and assess current industry tools, practices and standards for developing complex system architectures (e.g. Modular Open Systems Architecture, Model Driven Architecture, etc.).
- Identify potential benefits to the Navy of shifting to evolving industry best practices.
- Recommend changes in Navy acquisition management, systems engineering, training, education, and business practices.
- Identify S&T investment paths.
- As appropriate, evaluate emerging tools for specifying, bidding, and engineering software-intensive systems and suggested strategies for use across multiple organizations.

Study Sponsor: Mr. Carl Siel, RDA (CHENG) and RDML Michael S. Frick, USN, PEO-IWS are co-sponsors for this study.

This page intentionally left blank

Appendix B



DEPARTMENT OF THE NAVY
OFFICE OF THE ASSISTANT SECRETARY
RESEARCH, DEVELOPMENT AND ACQUISITION
1000 NAVY PENTAGON
WASHINGTON DC 20350-1000

MAY 15 2006

MEMORANDUM FOR DISTRIBUTION

SUBJECT: Software Process Improvement Initiative

Successful development and acquisition of software is paramount for acquiring Naval Warfighting and business systems. There are many parallel and related efforts underway that address improvement in the acquisition of software products: mandates such as Public Law 107-314 Section 804 and the Clinger-Cohen Act; initiatives such as Software Assurance and Open Architecture (OA); and the development of best practice models such as the Capability Maturity Model Integration (CMMI) for Acquisition. To consolidate these efforts into a focused initiative, I have formed a steering group composed of my senior engineering professionals and led by the ASN (RD&A) Chief Engineer. This group will evaluate existing policies and implement process improvements to enhance our ability to develop and acquire software without sacrificing the cost, schedule and performance goals of our acquisition programs.

Additionally, five focus teams, led by department software engineering professionals, have been established to achieve our strategic software goals (see attachment):

- Software Acquisition Management (SAM) Focus Team
- Software Systems Engineering (SSE) Focus Team
- Software Development (SWDEV) Techniques Focus Team
- Business Implications Focus Team
- Human Resources Focus Team

To energize the process, I am initiating two projects immediately - software education and software acquisition discipline. These will initially be required for all ACAT I and II level acquisition programs.

The first project is to ensure key government program office personnel have a minimum level of knowledge of software acquisition and engineering management practices. The objective is to quickly establish a solid foundation of trained government software acquisition professionals. To enable this understanding, the following courses will be required for all government Program Managers, Deputy Program Managers, and Technical Directors/Chief Engineers assigned to an ACAT I or II program and must be completed within 18 months:

SUBJECT: Software Process Improvement Initiative

1. DAU course SAM 101 - Basic Software Acquisition Management (24 hour distance learning course)
2. SEI course - Introduction to CMMI (3 days of classroom training)

The second project is to ensure that software development efforts in software intensive system programs are conducted by contractors who have a software process improvement program established that addresses at a minimum:

Software Acquisition Planning
Requirements Development
Management
Project Management and Oversight
Risk Management.

These functional processes must be demonstrated and exercised by the developer in a continuous manner and be equivalent to that articulated by CMMI capability level 3. They will be assessed by ASN RDA Chief Engineer and the senior steering group on a periodic basis. Statements of Work for all applicable future procurements after 1 October, 2006 must address these requirements.

The development, acquisition, and delivery of software are key to the Navy's ability to successfully conduct its Warfighting and Business operations. I need your commitment and I most strongly encourage your support as we address this significant challenge.



Delores M. Etter

Attachment: As Indicated

SUBJECT: Software Process Improvement Initiative

Distribution:

COMNAVSEASYS
COMNAVAIRSYS
COMSPAWARSYS
MARCORSYS
PEO JSF
PEO T
PEO A
PEO W
PEO SHIPS
PEO SUBS
PEO CARRIERS
PEO IWS
PEO EIS
PEO C4I
PEO LMW
PEO SPACE
DRPM SSP
ASN (RD&A) CHENG

Copy to:

DASN ACQ MGT
DASN AIR
DASN C4I/SPACE
DASN IWS
DASN LMW
DASN LOG
DASN M&B
DASN RDT&E
DASN SHIPS
NAVY IPO
DACM
NAVSEA (05, 06)
NAVAIR (4.0)
SPAWAR (05)
COMNAVFACSYS
COMNAVSUPSYS

Software Process Improvement Initiative Teams

Software Acquisition Management (SAM) Focus Team – is chartered to adopt a standard software acquisition life cycle model, develop a tailor able organizational structure with roles and responsibilities, and establish a set of software events and products that will apply over the acquisition life cycle including earned value management system (EVMS) benchmarks for tracking software development progress. This team will also draft the software policy documents with input from the other four teams for submission to the senior steering group.

Software Systems Engineering (SSE) Focus Team – is chartered to integrate software engineering events and products into traditional systems engineering practices. This team will establish compliance methodologies, develop a tailorable set of software metrics, and examine the use of a software systems engineering plan (SSEP) as a means to institutionalize these software practices for presentation at milestone decision points.

Software Development (SWDEV) Techniques Focus Team – is chartered to research and evaluate current and emerging software development methodologies and their supporting standards, to understand their positive and negative attributes, and determine how they could be applied to enhance our software development and acquisition activities. This team will interface with FFRDC's, government labs, academia, and industrial communities in searching out these potentially innovative methodologies.

Business Implications Focus Team – is chartered to examine our business, acquisition, and contracting strategies and practices to ensure the Navy is a "smart buyer" of software products whether they are "off the shelf" purchases or sponsored developments. This should include a fundamental understanding of the implications of emergent software development techniques upon our internal practices and the potential ramifications upon our industrial base. This group will also be expected to develop standardized contract language for the software procurement and development efforts.

Human Resources Focus Team – is chartered to refine the required skills and capabilities needed by government software acquisition and engineering professionals, and to identify a required set training courses tailored to the respective roles and responsibilities of these professionals.

Attachment

Appendix C

Acronym List

<u>Acronym</u>	<u>Definition</u>
ABM/GIFC	Advanced Battle Manager/Global Integrated Fire Control
Ada	The name of the DOD programming language mandated in the 1980s. Not an acronym, the language was named after August Ada, Countess of Lovelace, purported to be the first programmer.
ARCI/APB	Acoustic Rapid COTS Insertion/Advanced Processor Build
ASSIP	Army's Strategic Software Improvement Program
BAMS	Broad Area Maritime Surveillance
CEC	Cooperative Engagement Capability
CG(x)	A new Cruiser class
CHENG	Chief Engineer
CJCS	Chairman of the Joint Chiefs of Staff
CMMI	Capability Maturity Model Integration
COTS	Commercial Off-the-Shelf
DAWIA	Defense Acquisition Workforce Improvement Act
DSB	Defense Science Board
DSL	Domain Specific Language
FCS	(Army) Future Combat System
FFRDC	Federally Funded Research and Development Center
FORCEnet	"Maritime Framework for the GIG"
GAO	Government Accountability Office
GIG	Global Information Grid
GSA	General Services Administration
IPAs	Intergovernmental Personnel Act
JSF	Joint Strike Fighter (F-35)
JSSEO	Joint SIAP System Engineering Organization
JTRS	Joint Tactical Radio System
LCS	Littoral Combat Ship
LMRS	(Navy) Long-term Mine Reconnaissance System (replaced NMRS)

MDA ®	Model Driven Architecture ®
MDD™	Model Driven Development™
MSLOC	Million Single Lines of Code
NSA	National Security Agency
NSS	National Security System
QDR	Quadrennial Defense Review
RESET	Rapid Evolution Software Engineering Teams
SBIR	Small Business Innovative Research
SEI	Software Engineering Institute
SIAP	Single Integrated Air Picture
SLOC	Single Lines of Code
SOA	Service Oriented Architecture
STARS	Software Technology for Adaptive Reliable Systems
UARC	University Affiliated Research Center
ULS	Ultra Large Systems
UML	Unified Modeling Language